La Biblioteca di Atlantide

Quaderni di letteratura grigia

www.sergiofumich.com

Ca' "La Gatera"
26822 Brembio (LO) – Via Togliatti, 3
www.kerauniaware.com

SERGIO FUMICH

LOGICA E µPROCESSORI

NOTE AD USO DIDATTICO CON I PRIMI RUDIMENTI
DELL'ALGEBRA DI BOOLE

Ca' "La Gatera"
2006

Algebra di Boole

Un insieme **B = {a,b,c,...}** sul quale sono definite le operazioni binarie + e ○ si chiama algebra di Boole se per esso valgono i seguenti assiomi:

1. Le operazioni binarie + e ○ godono della proprietà commutativa, cioè è:

$a + b = b + a \qquad a \circ b = b \circ a.$

2. L'insieme **B** ha due elementi particolari: un primo elemento **0** ed un ultimo elemento **1** tali che sia:

$a + 0 = a \qquad a \circ 1 = a.$

3. Ciascuna operazione gode della proprietà distributiva rispetto all'altra, cioè è:

$a + (b \circ c) = (a + b) \circ (a + c)$

$a \circ (b + c) = (a \circ b) + (a \circ c).$

4. Per ogni elemento **a** che appartiene all'insieme **B** esiste un elemento complementare **ā** tale che:

$a + \bar{a} = 1 \qquad a \circ \bar{a} = 0.$

La simmetria dei postulati rispetto alle operazioni + e ○ e rispetto agli elementi **0** e **1** giustifica il seguente *principio di dualità*:

Ogni relazione deducibile dai postulati di un'algebra di Boole rimane valida se si scambiano tra loro i simboli + e ○ e gli elementi particolari **0** e **1**.

Come conseguenza basterà dimostrare solo una

fra due relazioni duali.

Dimostriamo che, per ogni **a** che appartiene a **B,** è: $\mathbf{a} + \mathbf{a} = \mathbf{a}$ e $\mathbf{a} \circ \mathbf{a} = \mathbf{a}$.

È: $a + a = (a + a) \circ 1 = (a + a) \circ (a + \bar{a}) = a + (a \circ \bar{a}) = a + 0 = a$. Per il principio di dualità è vera anche la relazione duale.

Dimostriamo che, per ogni **a** che appartiene a **B**, è: $\mathbf{a} + \mathbf{1} = \mathbf{1}$ e $\mathbf{a} \circ \mathbf{0} = \mathbf{0}$.

È: $a \circ 0 = 0 + (a \circ 0) = (a \circ \bar{a}) + (a \circ 0) = a \circ (\bar{a} + 0) = a \circ \bar{a} = 0$. Per il principio di dualità è vera anche la relazione duale.

Dimostriamo che, per ogni **a**, **b** appartenenti a **B**, è: $\mathbf{a} + (\mathbf{a} \circ \mathbf{b}) = \mathbf{a}$ e $\mathbf{a} \circ (\mathbf{a} + \mathbf{b}) = \mathbf{a}$.

È: $a + (a \circ b) = (a \circ 1) + (a \circ b) = a \circ (1 + b) = a \circ 1 = a$. Per il principio di dualità è vera anche la relazione duale.

Valgono le seguenti relazioni:

Proprietà associativa:

$(a + b) + c = a + (b + c)$, $(a \circ b) \circ c = a \circ (b \circ c)$.

Teoremi di De Morgan:

$\overline{a + b} = \bar{a} \circ \bar{b}$,

$\overline{a \circ b} = \bar{a} + \bar{b}$.

Dimostriamo che, per ogni **a**, **b** appartenenti a **B**, è: $\mathbf{a} + \bar{\mathbf{a}} \circ \mathbf{b} = \mathbf{a} + \mathbf{b}$.

È: $a + \bar{a} \circ b = a \circ (a + b) + \bar{a} \circ b = a \circ a + a \circ b + \bar{a} \circ b = a + b \circ (a + \bar{a}) = a + b \circ 1 = a + b$.

Se **B** è costituito da due soli elementi, $\mathbf{B} = \{\mathbf{a},\mathbf{b}\}$, è:

$$\mathbf{a} + \mathbf{b} = \mathbf{1}, \qquad \mathbf{a} \circ \mathbf{b} = \mathbf{0}.$$

Se **B** = {**a**,**b**,**c**} è: **a** + **b** + **c** = **1**, **a** ○ **b** ○ **c** = **0**.

Ogni espressione contenente un numero finito di elementi di un'algebra di Boole collegati dalle operazioni + e ○ si chiama *funzione di Boole*.

Il numero di variabili in ciascuna funzione è dato dal numero delle lettere distinte che in essa appaiono senza tenere conto se sono o non sono soprassegnate. Ad esempio:

$x \circ \overline{x}$ è funzione di una variabile;

$x \circ \overline{y}$ è funzione di due variabili.

Nel seguito ci limiteremo al caso **B** = {**0**,**1**}.

Le *tavole della verità* sono tabelle sulle quali si riportano tutte le combinazioni binarie delle variabili che si considerano ed i risultati delle operazioni che con esse si vogliono fare.

Funzione NOT: **f** = **ā**

a	**ā**
0	**1**
1	**0**

Funzione AND: **f** = **a** ○ **b**

a	**b**	**a** ○ **b**
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

Funzione OR: **f** = **a** + **b**

a	**b**	**a** + **b**
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

Funzione NAND: **f** = $\overline{\mathbf{a} \circ \mathbf{b}}$

a	**b**	$\overline{\mathbf{a} \circ \mathbf{b}}$
0	**0**	**1**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Funzione NOR: **f** = $\overline{\mathbf{a} + \mathbf{b}}$

a	**b**	$\overline{\mathbf{a} + \mathbf{b}}$
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**0**

Le tavole della verità sono il mezzo più semplice per dimostrare i teoremi dell'algebra booleana. Vediamo alcuni esempi.

Teorema della doppia inversione:

Una variabile invertita due volte riprende il suo valore primitivo: $\overline{\overline{\mathbf{a}}} = \mathbf{a}$.

$\mathbf{a}$	$\overline{\mathbf{a}}$	$\overline{\overline{\mathbf{a}}}$
0	**1**	**0**
1	**0**	**1**

Primo teorema di De Morgan: $\overline{\mathbf{a} \circ \mathbf{b}} = \overline{\mathbf{a}} + \overline{\mathbf{b}}$.

La dimostrazione è data dalla seguente tavola:

$\mathbf{a}$	$\mathbf{b}$	$\overline{\mathbf{a}}$	$\overline{\mathbf{b}}$	$\mathbf{a} \circ \mathbf{b}$	$\overline{\mathbf{a} \circ \mathbf{b}}$	$\overline{\mathbf{a}} + \overline{\mathbf{b}}$
0	**0**	**1**	**1**	**0**	**1**	**1**
0	**1**	**1**	**0**	**0**	**1**	**1**
1	**0**	**0**	**1**	**0**	**1**	**1**
1	**1**	**0**	**0**	**1**	**0**	**0**

Secondo teorema di De Morgan: $\overline{\mathbf{a} + \mathbf{b}} = \overline{\mathbf{a}} \circ \overline{\mathbf{b}}$.

La dimostrazione è data dalla seguente tavola:

$\mathbf{a}$	$\mathbf{b}$	$\overline{\mathbf{a}}$	$\overline{\mathbf{b}}$	$\mathbf{a} + \mathbf{b}$	$\overline{\mathbf{a} + \mathbf{b}}$	$\overline{\mathbf{a}} \circ \overline{\mathbf{b}}$
0	**0**	**1**	**1**	**0**	**1**	**1**
0	**1**	**1**	**0**	**1**	**0**	**0**
1	**0**	**0**	**1**	**1**	**0**	**0**
1	**1**	**0**	**0**	**1**	**0**	**0**

L’elemento base dell’attuale tecnologia elettronica digitale è il transistor. Supponiamo di avere un transistor NPN con l’emettitore collegato a massa ed il

collettore collegato a +**5**V attraverso un resistore. Se sulla base del transistor è presente tensione positiva, il transistor conduce e sul collettore è presente una tensione prossima a **0**V. Se al contrario sulla base è presente una tensione di **0**V, il transistor non conduce e quindi sul collettore vi sarà una tensione positiva di **5**V.

Si intuisce, dunque, il possibile uso logico di questo elemento: se consideriamo livelli di tensione prossimi a **0**V come **0** e livelli positivi superiori ad una certa soglia come **1**, il transistor, considerata come ingresso la base e come uscita il collettore, può essere usato per realizzare un *invertitore*, cioè un dispositivo che realizzi la funzione NOT.

Con due transistor, collegati in modo che l'emettitore dell'uno sia collegato con il collettore del secondo, si può realizzare la funzione NAND, e quindi usando il NOT e NAND si possono realizzare facilmente le altre funzioni, grazie al teorema della doppia inversione ed ai teoremi di De Morgan.

Avendo a disposizione dispositivi che realizzino le funzioni logiche, si può facilmente costruire un dispositivo *sommatore*. Se **x**, **y** sono le cifre binarie in ingresso a tale dispositivo, e **z** e **c** (*carry* o riporto) sono le cifre in uscita, si può scrivere la seguente tavola di verità:

x	**y**	**c**	**z**
0	**0**	**0**	**0**
0	**1**	**0**	**1**
1	**0**	**0**	**1**
1	**1**	**1**	**0**

Osserviamo che **c** può essere ottenuto realizzando un circuito che fornisca in uscita $\mathbf{x} \circ \mathbf{y}$, mentre **z** è dato dall'OR esclusivo, che può essere realizzato come $\mathbf{x} \circ \overline{\mathbf{y}} + \overline{\mathbf{x}} \circ \mathbf{y}$.

Applicando i teoremi di De Morgan si ha:

$$\mathbf{x} \circ \overline{\mathbf{y}} + \overline{\mathbf{x}} \circ \mathbf{y} = \overline{\overline{\mathbf{x} \circ \overline{\mathbf{y}}} \circ \overline{\overline{\mathbf{x}} \circ \mathbf{y}}}$$

ed è anche:

$$\mathbf{x} \circ \mathbf{y} = \overline{\overline{\mathbf{x} \circ \mathbf{y}}}$$

Osservando che $\overline{\mathbf{x} \circ \mathbf{x}} = \overline{\mathbf{x}}$**,** per realizzare il circuito che sommi due bit si possono usare 7 blocchi NAND.

BITS, BYTES, OPERATORI BOOLEANI

Il sistema di numerazione binario è la base per tutte le operazioni di un computer. Tale sistema richiede l'uso di soltanto due cifre o "digit" (secondo la dizione inglese): **0** e **1**. Queste due possibilità, **0** e **1**, possono essere facilmente rappresentate rispettivamente da un basso ed un alto livello di tensione.

I numeri decimali da 0 a 5 vengono scritti nel sistema binario come segue:

0 = 0

1 = 1

2 = 10

3 = 11

4 = 100

5 = 101

Come si intuisce, la rappresentazione binaria dei numeri, man mano che si procede lungo la serie naturale, richiede un numero di digit rapidamente crescente.

Ad esempio, il numero **1024** richiede 11 digit binari; il numero **65536** 17 digit binari:

1024 = 10000000000

65536 = 10000000000000000

Numero decimale	Numero binario	
0	0	
1	1	
2	10	
3	11	
4	100	
5	101	
6	110	
7	111	
8	1000	= 4 bit
15	1111	= 4 bit
16	10000	
31	11111	
32	100000	
63	111111	
64	1000000	
127	1111111	
128	10000000	= 8 bit
255	11111111	= 8 bit
256	100000000	
511	111111111	
512	1000000000	= 10 bit
1023	1111111111	= 10 bit
1024	10000000000	
2047	11111111111	
2048	100000000000	
4095	111111111111	
4096	1000000000000	
8191	1111111111111	
8192	10000000000000	
16383	11111111111111	
16384	100000000000000	
32767	111111111111111	
32768	1000000000000000	= 16 bit
65535	1111111111111111	= 16 bit

Nel sistema decimale ciascun digit rappresenta una potenza del **10**. Ad esempio:

$423 = 4 \cdot 100 + 2 \cdot 10 + 3 \cdot 1 = 4 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$

Nel sistema binario, ciascun digit rappresenta una potenza del **2**:

$1101 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1$

Pertanto, il numero binario **1101** è il numero decimale **13**.

La conversione tra i due sistemi di numerazione, binario e decimale, può essere attuata usando questi metodi; tuttavia difficilmente si convertono dal binario ed in binario numeri più grandi di **15**. Numeri maggiori sono convertiti piuttosto nel *sistema esadecimale*.

Il sistema esadecimale, o a base **16**, richiede 16 differenti digit. I digit dallo **0** al **9** sono presi in prestito dal sistema decimale, mentre per rappresentare gli altri 6 si usano le prime sei lettere dell'alfabeto; così si ha:

A = 10

B = 11

C = 12

D = 13

E = 14

F = 15

Nel sistema esadecimale ciascun digit rappresenta una potenza del **16**. Ad esempio:

$1D3 = 1 \cdot 16^2 + D \cdot 16^1 + 3 \cdot 16^0 = 1 \cdot 256 + 13 \cdot 16 + 3 \cdot 1$

Cioè, l’esadecimale **1D3** è il numero decimale **467**.

Per evitare confusioni, usando i tre sistemi sarà necessario distinguere le tre rappresentazioni. Là, dove vi può essere ambiguità, si può far seguire il numero da una lettera che indichi il sistema di rappresentazione:

esadecimale: 10H

decimale: 10D o semplicemente **10**

binario: 10B

La conversione di piccoli numeri può essere fatta usando la seguente tabella di conversione:

D	B	H
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

I numeri esadecimali possono essere convertiti nel sistema decimale facilmente, usando le potenze del **16**. Come esempio vediamo la conversione in decimale del numero **2AF3H**:

$$2AF3H = 2 \cdot 16^3 + A \cdot 16^2 + F \cdot 16^1 + 3 \cdot 16^0 =$$

$$= 2 \cdot 4096 + 10 \cdot 256 + 15 \cdot 16 + 3 \cdot 1 =$$

$$= 8192 + 2560 + 240 + 3 = 10995$$

Le conversioni dal decimale all'esadecimale possono essere fatte usando la divisione e la seguente tabella di potenze del **16**:

16^0 =	**1**
16^1 =	**16**
16^2 =	**256**
16^3 =	**4096**
16^4 =	**65536**

Numeri più grandi di **16^4** si incontrano raramente.

Nella conversione inizieremo sempre col dividere il numero con la più grande potenza di **16** possibile. Come esempio riconvertiamo il numero **10995** in esadecimale:

10995	4096				
2803	2 →	**2**			
	2803	256			
	243	10 →	**A**		
		243	16		
		3	15 →	**F**	
			3	1	
			0	3 →	**3**

La conversione esadecimale-binario è facilissima, poiché ogni digit esadecimale può essere tradotto in 4 digit binari usando la tabella di conversione. Per esempio:

53DH = 0101 0011 1101 = 10100111101B

I numeri binari possono essere convertiti in esadecimale dividendo i digit in gruppi di quattro partendo da destra e convertendo ciascun gruppo usando la tabella:

1011101101B = 0010 1110 1101 = 2EDH

Un *bit* è l'unità elementare di informazione memorizzabile. Indicheremo in seguito un bit racchiudendo il valore **0** o il valore **1** in un quadratino:

0	o	**1**

Talvolta è importante considerare un solo bit, ma generalmente un'informazione è rappresentata da una serie di bit.

Un *byte* è un insieme di 8 bit adiacenti:

1	**0**	**1**	**0**	**0**	**1**	**0**	**1**

Per individuare facilmente i singoli bit nell'ambito di un byte, i bit sono numerati da **0** a **7**, da destra a sinistra:

1	**0**	**1**	**0**	**0**	**1**	**0**	**1**
7	6	5	4	3	2	1	0

Il bit più significativo (MSB, most significant bit) è il bit 7. Il bit meno significativo (LSB, least signifi-

cant bit) è il bit 0.

Poiché quattro bit corrispondono ad un digit esadecimale, si può rappresentare il contenuto di un byte con due digit esadecimali. Talvolta poi, un gruppo di quattro bit è chiamato *nibble*.

Il byte è l'unità aritmetica base di un microcomputer. Il più grande numero rappresentabile con un byte è **255**:

1	**1**	**1**	**1**	**1**	**1**	**1**	**1**

= FFH = 255

Con un byte, dunque, si possono rappresentare numeri positivi fino al valore **255**. Solitamente, però, ci interessa anche la possibilità di rappresentare sia numeri positivi che numeri negativi. Si può ottenere questo riservando uno degli otto bit come indicatore del segno del numero:

0	**X**	**X**	**X**	**X**	**X**	**X**	**X**

contenuto positivo (bit di segno **0**)

1	**X**	**X**	**X**	**X**	**X**	**X**	**X**

contenuto negativo (bit di segno **1**).

Riservando uno degli otto bit per indicare il segno, ci rimangono soltanto sette bit per rappresentare il valore del numero. Pertanto il più grande numero rappresentabile è 127:

0	**1**	**1**	**1**	**1**	**1**	**1**	**1**

= 7FH = 127

Fatta tale convenzione, un primo modo semplice

per rappresentare i numeri negativi è quello di complementare semplicemente il segno lasciando immodificato il valore. Sebbene questo sia un metodo facile da capire, tuttavia è di uso difficoltoso.

Un secondo metodo di rappresentazione potrebbe essere quello di complementare tutti i bit (fare cioè il complemento a **1**):

0	**0**	**0**	**1**	**1**	**0**	**1**	**0**	= 26

1	**1**	**1**	**0**	**0**	**1**	**0**	**1**	=

= complemento a 1 di 26.

Tuttavia anche questo metodo di rappresentazione ha un difetto: sicuramente utilizzando numeri positivi e negativi vogliamo che, sommando **26** con **–26** il risultato sia zero. Ma:

	0	**0**	**0**	**1**	**1**	**0**	**1**	**0**
+	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**
=	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

che non è certo tutti **0**.

Tuttavia, se aggiungiamo al risultato 1 e ignoriamo il riporto (carry), abbiamo proprio ciò che volevamo.

		1	**1**	**1**	**1**	**1**	**1**	**1**	**1**
+		**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
	carry								
=	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Da queste considerazioni deriva la rappresentazione dei numeri negativi in *complemento a* **2**.

Il complemento a **2** di un numero si ottiene aggiungendo **1** al suo complemento a **1**:

26	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**0**
complemento a 1 di 26	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**
+	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
=	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**

Il risultato rappresenta il complemento a **2** di **26**.

Lo stesso metodo può essere usato per convertire un numero negativo nel suo opposto.

Possiamo ora eseguire addizioni e sottrazioni di numeri binari a 8 bit.

11 + 7 = 18

0	**0**	**0**	**0**	**1**	**0**	**1**	**1**	
0	**0**	**0**	**0**	**0**	**1**	**1**	**1**	+
0	**0**	**0**	**1**	**0**	**0**	**1**	**0**	=

11 − 7 = 4

0	**0**	**0**	**0**	**1**	**0**	**1**	**1**	
1	**1**	**1**	**1**	**1**	**0**	**0**	**1**	+
0	**0**	**0**	**0**	**0**	**1**	**0**	**0**	=

Abbiamo già accennato ad uno dei flag aritmetici, il flag di carry (riporto). Il carry è posto a **1** ogniqualvolta il risultato di una somma è più grande di otto bit.

		carry								
	-1		**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
+	1		**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
=		**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Nelle somme il bit di carry è spesso trascurato. È semplicemente un sottoprodotto della rappresentazione in complemento a **2**.

Nelle sottrazioni il bit di carry ha maggior significato.

		carry								
	1		**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
−	2		**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
=		**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

1 − 2 = **−1**: è il caso in cui interviene un prestito. Pertanto il bit di carry ci dice che abbiamo sottratto un numero da un numero più piccolo.

Il flag di overflow ci avverte invece se qualcosa è andato storto. Sappiamo che un numero di otto bit con un bit di segno e sette bit di valore non può essere più grande di **127**. Il numero più piccolo è **−128**:

1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	= -128.

Supponiamo di sommare **75** e **80**:

75	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**
+ 80	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**
= 155	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**

È successo che il bit più significativo del risultato è stato riportato nel bit di segno facendo sì che il risultato appaia come un numero negativo. Il flag di overflow viene in questo caso posto uguale a **1** per indicare che il risultato non è valido. Questo può accadere anche quando si sommano due grandi (in valore assoluto) numeri negativi:

-75		**1**	**0**	**1**	**1**	**0**	**1**	**0**	**1**
+ (-80)	**carry**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**0**
= -155	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**

Di nuovo il bit di segno è stato complimentato e di conseguenza anche il flag di overflow viene posizionato a **1**. In questo caso anche il flag di carry è **1**, ma questo fatto ora non interessa.

Se sono sommati due operandi di segno diverso, non può esserci overflow.

Il flag di *segno* indica se il risultato di una operazione aritmetica è positivo o negativo. È una copia esatta del bit di segno. Pertanto è **0** quando il risultato è positivo, **1** quando il risultato è negativo.

Poiché un risultato zero ha uno zero nel bit di segno, è ovviamente considerato positivo. Un risultato zero, tuttavia, è così importante che è riportato in

un flag particolare: il flag *zero* è posizionato a **1** quando il risultato è zero.

Sommario dei flag nell'aritmetica a **8** bit:

Carry (C): **1** quando c'è un riporto oltre l'ottavo bit;
0 altrimenti.

Overflow (W): **1** quando interviene un overflow;
0 altrimenti.

Segno (S): **1** se il risultato è negativo;
0 altrimenti.

Zero (Z): **1** se il risultato è zero;
0 altrimenti.

Su coppie di byte, oltre che in modo aritmetico, si può operare in modo logico con operatori logici e booleani. Gli operatori booleani trattano tutti gli otto bit nello stesso modo, anche il bit di segno.

AND

L'operatore AND compara due byte per bit. Se un bit è a **1** in entrambi gli operandi, è posizionato a **1** anche nel risultato. Altrimenti il bit del risultato è posto a **0**.

Tavola di verità:

A	**B**	**A AND B**
1	**1**	**1**
1	**0**	**0**
0	**1**	**0**
0	**0**	**0**

Esempio:

A	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**1**
B	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**1**
A AND B	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

OR

L'operatore OR compara due byte bit per bit. Se un bit è a **1** in almeno uno dei due operandi, esso è posizionato a **1** nel risultato. Altrimenti nel corrispondente bit del risultato è messo uno **0**.

Tabella di verità:

A	**B**	**A OR B**
1	**1**	**1**
1	**0**	**1**
0	**1**	**1**
0	**0**	**0**

Esempio:

A	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**1**
B	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**1**
A OR B	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**1**

XOR

L'operatore XOR (OR esclusivo) compara due by-

te bit per bit. Se il bit è a **1** in almeno uno dei due operandi, ma non in entrambi, il corrispondente bit del risultato è posto a **1**, altrimenti a **0**.

Tabella di verità:

A	**B**	**A XOR B**
1	**1**	**0**
1	**0**	**1**
0	**1**	**1**
0	**0**	**0**

Esempio:

A	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**1**
B	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**1**
A XOR B	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**

Il flag **carry** è resettato (messo a **0**) da AND, OR, XOR. Il flag **zero** che indica quando tutti i bit del risultato sono a zero, è a **1** se il risultato è zero, a **0** altrimenti. Il flag **segno** riporta il valore del bit più significativo del risultato come nelle operazioni aritmetiche.

Le operazioni logiche interessano il flag di **parità** invece del flag di **overflow**. Il flag di **parità** indica se un numero pari o dispari di bit sono nel risultato settati (posizionati a **1**). Il flag di **parità** è settato quando un numero pari di bit sono a **1** nel risultato, altrimenti esso è resettato.

Risultato:	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**0**

5 bit posti a **1**: flag di parità P a **0**.

Risultato:	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

2 bit posti a **1**: flag di parità P a **1**.

COLLANA *"La Biblioteca di Atlantide"*

Collana di letteratura grigia: scienze matematiche, fisiche e naturali, informatica e telematica, linguaggi, arte e letteratura, storia e filosofia, varia umanità.

Opuscoli pubblicati:

Sergio Fumich, *Numerazione e codici. Appunti di matematica applicata al calcolatore.*

Sergio Fumich, *I numeri naturali. Appunti di matematica elementare dal punto di vista superiore.*

NOTIZIA

Sergio Fumich è nato a Trieste nel 1947. Dal 1970 si è trasferito a Brembio, piccolo comune del Lodigiano. Lavora a Milano presso un'importante Fondazione lombarda che si occupa di formazione professionale. Ha svolto attività pubblicistica dal 1978 al 1995 come collaboratore del quotidiano di Lodi *Il Cittadino*, come direttore responsabile di alcuni fogli locali e della rivista di poesia *Keraunia*. Ha pubblicato libri di poesia e di racconti e opuscoli divulgativi.

Ca' "La Gatera"

Edizione fuori commercio
Finito di stampare a Brembio (LO) con tecniche elettroniche nel febbraio 2006
Di questo opuscolo sono stati tirati 200 esemplari.